

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**

Integrating Temporal, Real-Time, and Active Databases - Ramamritham.. (1996) (Correct) (3 citations)  
Integrating temporal, real-time, and active databases Krithi Ramamritham, Juvisankar, Han

[www.ccs.cs.umass.edu/~sim/sigrec96.ps](http://www.ccs.cs.umass.edu/~sim/sigrec96.ps)

Can High Bandwidth and Latency Justify Large Cache Blocks.. - Bianchini, LeBlanc (1994) (Correct) (1 citation)

multiprocessors use hardware caches to keep **data** close to the processors that need it, and thereby behavior of application programs and the remote **access** bandwidth and latency of the machine. Several the performance of coherent caches in shared-memory multiprocessors is the choice of block size.  
ftp.cs.rochester.edu/pub/papers/systems/94.tr486.Can\_high\_bandwidth\_and\_latency\_justify\_large\_cache\_blocks.ps.Z

Efficient Detection of All Pointer and Array Access Errors - Austin, Breach, Sohi (1993) (Correct) (23 citations)

overheads range from 130% to 540% with text and **data** size overheads typically below 100% c fl 1993

Efficient Detection of All Pointer and Array Access Errors Todd M. Austin Scott E. Breach Gurindar

ftp.cs.wisc.edu/tech-reports/reports/93/tr1197.ps.Z

Dynamic Access Ordering for Symmetric Shared-Memory Multiprocessors - McKee (1994) (Correct)

to the poor temporal and spatial locality of their **data** accesses. Moreover, the nature of memories

Dynamic Access Ordering for Symmetric Shared-Memory

ftp.cs.virginia.edu/pub/techreports/CS-94-14.ps.Z

Memory Scalability in Constraint-Based Multimedia Style.. - Cumararatunge, Munson (1998) (Correct)

syntax trees are very large and the constraint **data** for a medium-sized source file can easily consume is constraint by its left sibling's Y attribute. AccessOp(Y) LeftSib )VertPos:Y Figure 3: Inverse path April 1998 (provisional paper :July 27, 1997) Memory Scalability in Constraint-Based Multimedia Style  
www.cs.uwm.edu/faculty/munson/pubs/ep98.ps

Table-Lookup Approach for Compiling Two-Level Data-Processor.. - Kuei-Ping Shih (1997) (Correct)

Table-Lookup Approach for Compiling Two-Level Data-Processor Mappings in HPF Kuei-Ping Shih y

xp1.csie.ncu.edu.tw/ftp/pub/tech-report/1997/./steven-LCPC97f.ps.gz

Correction of a Memory Management Method for Lock-Free Data.. - Michael, Scott (1995) (Correct) (5 citations)

of a Memory Management Method for Lock-Free Data Structures Maged M. Michael Michael L. Scott

structures, processes have to synchronize their **access** to them. Mutual exclusion locks are the most

Correction of a Memory Management Method for Lock-Free Data Structures

hypatia.dcs.qmw.ac.uk/data/edu/cs.rochester.edu/systems/95.tr599.Memory\_management\_for\_lock-free\_data\_structures.ps.g

Fortran 90D/HPF Compiler for Distributed Memory.. - Bozkus.. (1993) (Correct) (2 citations)

April 1, 1993 Abstract Fortran 90D/HPF is a **data** parallel language with special directives to distribution and communication for non-local **data** access. There has been significant research in Fortran 90D/HPF Compiler for Distributed Memory MIMD Computers: Design, Implementation, and

ftp.cis.ufl.edu/pub/faculty/ranka/compiler\_sc93.ps.Z

Shared Memory NUMA Programming on I-WAY - Nieplocha, Harrison (1996) (Correct) (8 citations)

(NUMA) programming model [1-3] and reference **data** in blocks to increase **data** locality in order to the Global Array shared-memory nonuniform memory-access programming model is explored on the I-WAY, 1 Shared Memory NUMA Programming on I-WAY J. Nieplocha and R. J.

ftp.pnl.gov/pub/permanent/global/iway.ps.Z

Effectiveness of Message Strip-Mining for Regular and.. - Akiyoshi Wakatani (1994) (Correct) (2 citations)

implement parallel algorithms by distributing large **data** structures across a multicomputer system. To hide (regular communication) and executor for indirect **access** (irregular communication) and have achieved to make a program executable on any distributed memory multicomputer. HPF also allows use of expensive

www.cse.ogi.edu/Sparse/paper/wakatani.pdcs.94.ps

First 20 documents [Next 20](#)

Try your query at: [Amazon](#) [Barnes & Noble](#) [Google \(RI\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

CiteSeer.IST - Copyright [NEC](#) and [IST](#)

Find: 

Documents

Citations

Searching for PHRASE **data access memory usage less frequent**.

Restrict to: [Header](#) [Title](#) Order by: [Expected citations](#) [Hubs](#) [Usage](#) [Date](#) Try: [Amazon](#) [B&N](#) [Google \(RI\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

No documents match Boolean query. Trying non-Boolean relevance query.

1000 documents found. Retrieving documents... Order: relevance to query.

[CC++, pC++, Charm++ and Orca: Languages for Parallel Programming - Niemiec \(1993\)](#) (Correct)  
in parallel computing. The need for functional and **data** parallelism is recognized and specifically error. When a currently executing thread wants to **access** a **data** unit of type sync which is not defined, it and the given support for distributed **memory** machines. Because of these criteria, concurrent  
[www.npac.syr.edu/projects/hpsin/doc/ccpp.ps](http://www.npac.syr.edu/projects/hpsin/doc/ccpp.ps)

[An Analytical Approach to File Prefetching - Lei \(1997\)](#) (Correct) (33 citations)  
hopefully in advance of the actual need for the **data**. Prefetched **data** is then placed in the client's is an effective technique for improving file **access** performance. In this paper, we present a file  
[www.mcl.cs.columbia.edu/papers/usenix97.ps.gz](http://www.mcl.cs.columbia.edu/papers/usenix97.ps.gz)

[Discovery of numerical dependencies in form of rational.. - Kiselev, Arseniev](#) (Correct)  
class [1, 2] Searching for structure hidden in **data** PolyAnalyst builds and tests hypotheses about primitives. Information in **databases** is **accessed** via special **data access** primitives which also compromise is achieved due to choice of more or **less** narrow set of formulae in which search is  
[www.megaputer.com/DOWN/PAismis6.PS](http://www.megaputer.com/DOWN/PAismis6.PS)

[The RD13 DAQ System and the Object Management Workbench - Bob Jones](#) (Correct)  
as used in the RD13 research project for **data** acquisition systems of high energy physics  
[rd13doc.cern.ch/public/doc/postscript/RD13\\_summerSchool95.Jones.ps](http://rd13doc.cern.ch/public/doc/postscript/RD13_summerSchool95.Jones.ps)

[Scheduling Access To Temporal Data In Real-Time Databases - Xiong, Sivasankaran.. \(1997\)](#) (Correct) (3 citations)  
1 Scheduling **Access** To Temporal Data In Real-Time Databases Ming Xiong, Rajendran  
[www-ccs.cs.umass.edu/~sim/rtdb-chapter96.ps](http://www-ccs.cs.umass.edu/~sim/rtdb-chapter96.ps)

[Mechanisms and Interfaces for Software-Extended Coherent Shared.. - Chaiken \(1994\)](#) (Correct) (3 citations)  
to optimize **accesses** to widely-shared, read-only **data** and improves one benchmark's performance by 22%  
[ftp.cag.lcs.mit.edu/pub/papers/chaiken-dissert-1-10.ps.Z](http://ftp.cag.lcs.mit.edu/pub/papers/chaiken-dissert-1-10.ps.Z)

[Cautious, Machine-Independent Performance Tuning for.. - Talbot, Bennett, Kelly](#) (Correct)  
to ensure that CPUs do not use stale cached **data**. In addition to the overheads of maintaining links. These all conspire to increase **memory access** times, and hence slow down the execution time of Machine-Independent Performance Tuning for Shared-Memory Multiprocessors Sarah A. M. Talbot, Andrew J.  
[www-ala.doc.ic.ac.uk/~phjk/Publications/CautiousMachineIndependent..EuroPar97.ps.gz](http://www-ala.doc.ic.ac.uk/~phjk/Publications/CautiousMachineIndependent..EuroPar97.ps.gz)

[Energy-Efficient Index Replication for Wireless Data Broadcasting - Yon Dohn](#) (Correct)  
Energy-Efficient Index Replication for Wireless **Data** Broadcasting Yon Dohn Chung Myoung Ho Kim  
all **data** stream must be read from the time of **access** request to the time until all requested **data** are  
[dbserver.kaist.ac.kr/NEW/warehouse/./thesis\\_store/ycchung7.ps.gz](http://dbserver.kaist.ac.kr/NEW/warehouse/./thesis_store/ycchung7.ps.gz)

[Using Data Structures within Genetic Programming - Langdon](#) (Correct) (5 citations)  
1 Using **Data** Structures within Genetic Programming W. B.  
is written so that it be independent of **memory access** implementation details. This is achieved by using some cases, provision of appropriately structured **memory** can indeed be advantageous to GP in comparison  
[ftp.cs.bham.ac.uk/pub/authors/W.B.Langdon/papers/WBL\\_gp96.submitted.ps](http://ftp.cs.bham.ac.uk/pub/authors/W.B.Langdon/papers/WBL_gp96.submitted.ps)

[Fine-Grain Dataflow Model And Algorithms For Visualization Systems - Song \(1994\)](#) (Correct)  
Fine-Grain **Dataflow** Model And Algorithms For Visualization  
[ftp.ncsa.uiuc.edu/ncsapubs/preprints/TR018.ps.Z](http://ftp.ncsa.uiuc.edu/ncsapubs/preprints/TR018.ps.Z)

[On using Network Memory to Improve the Performance of ... - Ioannidis, Markatos, ... \(1997\)](#) (Correct)  
ranging from CAD environment to large-scale **databases**. Unfortunately, adding transaction support to  
[www.ics.forth.gr/arch-vlsi/OS/papers/1997.TR190.Remote\\_memory\\_RVM.ps.gz](http://www.ics.forth.gr/arch-vlsi/OS/papers/1997.TR190.Remote_memory_RVM.ps.gz)

[Asynchronous Version Advancement in a Distributed.. - Jagadish, Mumick..](#) (Correct)  
Version Advancement in a Distributed Three Version **Database** H. V. Jagadish AT&T Laboratories  
[www.research.att.com/~mish/multiversi/n/synchronizing.ps.gz](http://www.research.att.com/~mish/multiversi/n/synchronizing.ps.gz)

Extending Locking Techniques to Improve Concurrent Database.. - Cesar Galindo-Legaria (Correct)  
 Extending locking techniques to improve concurrent **database access** Cesar Galindo-Legaria Fausto  
 locking techniques to improve concurrent **database access** Cesar Galindo-Legaria Fausto Rabitti IEI-CNR  
[ftp.inria.fr/associations/ERCIM/research\\_reports/ps/0495R036.ps](ftp.inria.fr/associations/ERCIM/research_reports/ps/0495R036.ps)

A Host Interface to the DTM Network - Ahlgren, Pink, Lindgren.. (1992) (Correct) (1 citation)  
 segmenting and reassembling packets to and from the **data** units of the dtm. The software part of the  
 port **memory** residing on the interface card and **accessible** over the SBus from the host cpu. The host  
 The interface is based on a dual port **memory** residing on the interface card and **accessible**  
<ftp.sics.se/pub/SICS-reports/Reports/SICS-R-92-01-SE.ps.Z>

The Zebra Striped Network File System - Hartman, Ousterhout (1993) (Correct) (118 citations)  
 system that increases throughput by striping file **data** across multiple servers. Rather than striping each  
[www.cs.arizona.edu/people/jhh/papers/zebra\\_tocs.ps](http://www.cs.arizona.edu/people/jhh/papers/zebra_tocs.ps)

Wait-Free Synchronization - Herlihy (1993) (Correct) (100 citations)  
 A wait-free implementation of a concurrent **data** object is one that guarantees that any process can  
 others, and some **memory** locations may be slower to **access**. A wait-free implementation of a concurrent **data**  
 may be inherently faster than others, and some **memory** locations may be slower to **access**. A wait-free  
[www.cs.brown.edu/courses/cs196a/topias.ps](http://www.cs.brown.edu/courses/cs196a/topias.ps)

Understanding Language Support for Irregular Parallelism - Raghavachari, Rogers (1995) (Correct) (1 citation)  
 Abstract. While software support for array-based, **data-parallel** algorithms has been studied extensively,  
 x4.2 Distribution User Control p p p Replication **Access** Control p ?D p p TSM a default)  
<ftp.cs.princeton.edu/techreports/1996/506.ps.Z>

Multi-level Partitioning and Scheduling under Local Memory.. - Qingyan Wang (1995) (Correct)  
 and DSP applications. Due to the large amount of **data** handled by such applications, the optimization of  
[www.cse.nd.edu/pub/Reports/1995/tr-95-11.ps.gz](http://www.cse.nd.edu/pub/Reports/1995/tr-95-11.ps.gz)

Implementation and Evaluation of Prefetching in the.. - Arunachalam.. (1996) (Correct)  
 be addressed to a certain extent, if the necessary **data** can be fetched from the disk before the I/O call  
 that does a considerable amount of disk **accesses**. A major portion of the compute processors'  
 three i860 processors) and 16 MBytes or more of **memory**. The nodes can operate individually or as a group  
[www.ece.nwu.edu/~meena/papers/ipps.ps](http://www.ece.nwu.edu/~meena/papers/ipps.ps)

On Using Intelligent Network Interface Cards to.. - Fluczynski, Martin,.. (1998) (Correct) (2 citations)  
 on the network interface, and transfers video **data** arriving from the network directly to the region  
 address space as the firmware, with low-overhead **access** to services and hardware, and don't require the  
 the network directly to the region of frame buffer **memory** representing the applications window. As a result  
[www.cs.berkeley.edu/~rmartin/papers/mef-nosssdav98.ps](http://www.cs.berkeley.edu/~rmartin/papers/mef-nosssdav98.ps)

[Documents 21 to 40](#) [Previous 20](#) [Next 20](#)

Try your query at: [Amazon](#) [Barnes & Noble](#) [Google \(RI\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

CiteSeer.IST - Copyright [NEC](#) and [IST](#)



Find:

Documents

Citations

Searching for PHRASE **less frequently occurring**.

Restrict to: [Header](#) [Title](#) Order by: [Expected citations](#) [Hubs](#) [Usage](#) [Date](#) Try: [Amazon](#) [B&N](#) [Google \(RI\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

21 documents found. Order: number of citations.

[Mining Frequent Patterns without Candidate Generation - Han, Pei, Yin \(1999\) \(Correct\) \(56 citations\)](#)  
have better chances of sharing nodes than **less frequently occurring** ones. Second, an FP-tree-based pattern  
<ftp.fas.sfu.ca/pub/cs/han/pdf/sigmod00.pdf>

[Random Texts Exhibit Zipf's-Law-Like Word Frequency Distribution - Li \(1992\) \(Correct\) \(15 citations\)](#)  
to have larger frequencies than those **less frequently occurring**. Nevertheless, it seems to be a puzzle  
<linkage.rockefeller.edu/wli/pub/zipf.ps>

[A Code Compression System Based on Pipelined Interpreters - Hoogerbrugge.. \(1999\) \(Correct\) \(8 citations\)](#)  
occurring codewords in fewer bits than **less frequently occurring** codewords. The best known statistical  
<einstein.et.tudelft.nl/~janh/philips-publications/compact.paper.pdf>

[Missing Feature Theory In ASR: Make Sure You Miss The ... - de Veth, de Wet.. \(1999\) \(Correct\) \(2 citations\)](#)  
are reliable estimators of the **less frequently occurring** feature values. As a consequence, it  
<www.dcs.shef.ac.uk/~ljupco/papers/deveth.1999.1.ps.gz>

[Expressive Probability Models For Speech Recognition And.. - Russell \(1999\) \(Correct\) \(1 citation\)](#)  
336 phonemes the last CDA result used **less frequently occurring** phonemes and had a size of 666. The  
<www.cs.berkeley.edu/~russell/papers/asru99-abstract.ps>

[Recovering Documentation-to-Source-Code Traceability Links.. - Marcus, Maletic \(Correct\)](#)  
of LSI, the term combinations which are **less frequently occurring** in the given document collection tend  
<trident.mcs.kent.edu/~amarcus/papers/icse03.pdf>

[Unknown - Algorithms For Lossless \(Correct\)](#)  
number of bits than the codes used for **less frequently occurring** symbols. Naturally occurring images  
<www.cise.ufl.edu/~sahni/papers/lossless23.pdf>

[Beauty is in the Genes of the Beholder - Harel, Unger, Sussman \(1984\) \(Correct\)](#)  
communication (1982) In contrast, in the **less frequently occurring** forms of DNA, i.e. A-DNA and Z-DNA, the  
<www.wisdom.weizmann.ac.il/~dharel/SCANNED.PAPERS/BeautyisInTheGenes.pdf>

[Discovering actionable patterns in event data - Hellerstein, Ma, Perng \(Correct\)](#)  
patterns that also address important but **less frequently occurring** phenomena. Furthermore, event data  
<researchweb.watson.ibm.com/journal/sj/413/hellerstein.pdf>

[Tsunami Generation By Submarine Mass Failure - Part II Case \(Correct\)](#)  
magnitude, with larger events typically **occurring less frequently** (Prior and Coleman, 1979 Edgers and  
with larger events typically **occurring less frequently** (Prior and Coleman, 1979 Edgers and Karlsrud,  
of these five mechanisms, in part because their **occurrence** is often concealed from view and in part  
[www.oce.uri.edu/~grilli/tsunami-asce\\_part2.pdf](www.oce.uri.edu/~grilli/tsunami-asce_part2.pdf)

[Unknown - The Process That \(Correct\)](#)  
and these are coded efficiently. The **less frequently occurring** patterns are coded in some less efficient  
[www.cs.colorado.edu/homes/jwilson/public\\_html/papers/data\\_compression.ps](www.cs.colorado.edu/homes/jwilson/public_html/papers/data_compression.ps)

[Organising keywords in a web search environment: a... - Ding, Chowdhury, Foo \(Correct\)](#)  
results were obtained if only the **less frequently occurring** terms were clustered and if the more  
[www.cs.vu.nl/~ying/download/ISKO\\_Finaldraft.pdf](www.cs.vu.nl/~ying/download/ISKO_Finaldraft.pdf)

[Automatic Speech Recognition in Adverse Acoustic Conditions - de Wet, de Veth, Cranen.. \(1999\) \(Correct\)](#)  
<challenge1.let.kun.nl/literature/dewet.1999.2.ps>

[Acoustic Pre-Processing For Optimal Effectivity Of.. - de Veth, Cranen, de.. \(1999\) \(Correct\)](#)  
<challenge1.let.kun.nl/literature/deveth.1999.3.ps>

[Variable Word Rate N-Grams - Gotoh, Renals \(Correct\)](#)  
constant word rate assumption) and the **less frequently occurring** 'C' 'less than 0.00029) In

[www.dcs.shef.ac.uk/~yg/.papers\\_ps2/istanbul.ps](http://www.dcs.shef.ac.uk/~yg/.papers_ps2/istanbul.ps)

Linearizable Read/Write Objects - Mavronicolas, Roth (Correct)

chosen in order, e.g. to degrade the **less frequently occurring** operation. A read operation simply  
[l2r.cs.uiuc.edu/~danr/Papers/linearJ.ps.gz](http://l2r.cs.uiuc.edu/~danr/Papers/linearJ.ps.gz)

Linguistically Engineered Tools for Speech Recognition.. - Van Ess-Dykema, Ries (1998) (Correct)

significant error reduction in the **less frequently occurring** Dialog Acts and we report on the  
[werner.ira.uka.de/papers/speech/ICSLP98/ICSLP98-carol.ps.gz](http://werner.ira.uka.de/papers/speech/ICSLP98/ICSLP98-carol.ps.gz)

Locating Special Events when Solving ODEs - Gladwell, Shampine (Correct)

event function, and is cautious in other **less frequently occurring** circumstances. If we did not insist on  
[cygnus.math.smu.edu/pub/gladwell/smevents.ps.gz](http://cygnus.math.smu.edu/pub/gladwell/smevents.ps.gz)

A Two stage process for accurate image segmentation - Campbell, Thomas, Troscianko (1997) (Correct)

cluster represents 'vegetation' Other, **less frequently occurring** labels, are also present but are  
[hypatia.dcs.qmw.ac.uk/data/uk/cs.bris.ac.uk/1997/1997-w-1.ps.gz](http://hypatia.dcs.qmw.ac.uk/data/uk/cs.bris.ac.uk/1997/1997-w-1.ps.gz)

*First 20 documents* [Next 20](#)

Try your query at: [Amazon](#) [Barnes & Noble](#) [Google \(RI\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

CiteSeer.IST - Copyright [NEC](#) and [IST](#)

Find: [Documents](#)[Citations](#)Searching for PHRASE **data access sequences**.Restrict to: [Header](#) [Title](#) Order by: [Expected citations](#) [Hubs](#) [Usage](#) [Date](#) Try: [Amazon](#) [B&N](#) [Google \(RI\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)

Order: number of citations.

[NS32FX161-15](#)[NS32FX161-20](#)[NS32FX164-20](#).. - General Description The [\(Correct\)](#)Processor Bus Cycles#65 3#5#5#8 **Data Access Sequences**#67 3#5#5#9 Bus Access Control[www.cs.utep.edu/~bdauriol/courses/Architecture/NSa.pdf](http://www.cs.utep.edu/~bdauriol/courses/Architecture/NSa.pdf)**One or more of the query terms is very common - only partial results have been returned. Try [Google \(RI\)](#).**Try your query at: [Amazon](#) [Barnes & Noble](#) [Google \(RI\)](#) [Google \(Web\)](#) [CSB](#) [DBLP](#)CiteSeer.IST - Copyright [NEC](#) and [IST](#)





US Patent &amp; Trademark Office

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide

hot cold path



THE ACM DIGITAL LIBRARY


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)
Terms used hot cold path

Found 3,610 of 132,857

Sort results by

relevance

Display results

expanded form

☒ [Save results to a Binder](#)
☒ [Search Tips](#)
☐ Open results in a new window

[Try an Advanced Search](#)
[Try this search in The ACM Guide](#)

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐

# 1 [Hot cold optimization of large Windows/NT applications](#)

Robert Cohn, P. Geoffrey Lowney

 December 1996 **Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture**

Full text available:

 pdf(1.14 MB) [Publisher Site](#)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

A dynamic instruction trace often contains many unnecessary instructions that are required only by the unexecuted portion of the program. Hot-cold optimization (HCO) is a technique that realizes this performance opportunity. HCO uses profile information to partition each routine into frequently executed (hot) and infrequently executed (cold) parts. Unnecessary operations in the hot portion are removed, and compensation code is added on transitions from hot to cold as needed. We evaluate HCO on a ...

**Keywords:** optimization, profile, NT, register allocation

# 2 [Software profiling for hot path prediction: less is more](#)

Evelyn Duesterwald, Vasanth Bala

 November 2000 **Proceedings of the ninth international conference on Architectural support for programming languages and operating systems**, Volume 34, 28 Issue 5, 5

Full text available: pdf(286.07 KB)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Recently, there has been a growing interest in exploiting profile information in adaptive systems such as just-in-time compilers, dynamic optimizers and, binary translators. In this paper, we show that sophisticated software profiling schemes that provide highly accurate information in an offline setting are ill-suited for these dynamic code generation systems. We experimentally demonstrate that hot path predictions must be made early in order to control the rising cost of missed opportunity tha ...

# 3 [Software profiling for hot path prediction: less is more](#)

Evelyn Duesterwald, Vasanth Bala

November 2000 **ACM SIGPLAN Notices**, Volume 35 Issue 11

Full text available: pdf(2.43 MB)

 Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

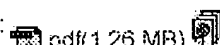
Recently, there has been a growing interest in exploiting profile information in adaptive systems such as just-in-time compilers, dynamic optimizers and, binary translators. In this paper, we show that sophisticated software profiling schemes that provide highly accurate information in an offline setting are ill-suited for these dynamic code generation systems. We experimentally demonstrate that hot path predictions must be made early in order to control the rising cost of missed opportunity tha ...

4 Compilation and run-time systems: Vacuum packing: extracting hardware-detected program phases for post-link optimization

Ronald D. Barnes, Erik M. Nystrom, Matthew C. Merten, Wen-mei W. Hwu

November 2002 **Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture**

Full text available:



[Publisher Site](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

This paper presents Vacuum Packing, a new approach to profile-based program optimization. Instead of using traditional aggregate or summarized execution profile weights, this approach uses a transparent hardware profiler to automatically detect execution phases and record branch profile information for each new phase. The code extraction algorithm then produces code packages that are specially formed for their corresponding phases. The algorithm compensates for the incomplete and often incoherent ...

5 Performance analysis of ATM Banyan networks with shared queueing—part II: correlated/unbalanced offered traffic

Achille Pattavina, Stefano Gianatti

August 1994 **IEEE/ACM Transactions on Networking (TON)**, Volume 2 Issue 4

Full text available: pdf(1.66 MB)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

6 Exploiting hardware performance counters with flow and context sensitive profiling

Glenn Ammons, Thomas Ball, James R. Larus

May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation**, Volume 32 Issue 5

Full text available: pdf(1.67 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

A program profile attributes run-time costs to portions of a program's execution. Most profiling systems suffer from two major deficiencies: first, they only apportion simple metrics, such as execution frequency or elapsed time to static, syntactic units, such as procedures or statements; second, they aggressively reduce the volume of information collected and reported, although aggregation can hide striking differences in program behavior. This paper addresses both concerns by exploiting the hardware ...

7 A hardware mechanism for dynamic extraction and relayout of program hot spots

Matthew C. Merten, Andrew R. Trick, Erik M. Nystrom, Ronald D. Barnes, Wen-mei W. Hwu

May 2000 **ACM SIGARCH Computer Architecture News , Proceedings of the 27th annual international symposium on Computer architecture**, Volume 28 Issue 2

Full text available: pdf(320.13 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper presents a new mechanism for collecting and deploying runtime optimized code. The code-collecting component resides in the instruction retirement stage and lays out hot execution paths to improve instruction fetch rate as well as enable further code optimization. The code deployment component uses an extension to the Branch Target Buffer to migrate execution into the new code without modifying the original code. No significant delay is added to the total execution of the program ...

8 Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels

Anindo Banerjee

August 1996 **ACM SIGCOMM Computer Communication Review , Conference proceedings on Applications, technologies, architectures, and protocols for computer communications**, Volume 26 Issue 4

Full text available: pdf(71.88 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


The paper presents a simulation study of the use of dispersity routing to provide fault

tolerance on top of a connection oriented realtime service such as that provided by the Tenet scheme. A framework to study the dispersity schemes is presented. The simulations show that the dispersity schemes, by dividing the connection's traffic among multiple paths in the network, have a beneficent effect on the capacity of the network. Thus, for certain classes of dispersity schemes, we obtain a small impr ...

#### 9 Online feedback-directed optimization of Java

Matthew Arnold, Michael Hind, Barbara G. Ryder

November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 37 Issue 11

Full text available:  pdf(463.00 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


This paper describes the implementation of an online feedback-directed optimization system. The system is fully automatic; it requires no prior (offline) profiling run. It uses a previously developed low-overhead instrumentation sampling framework to collect control flow graph edge profiles. This profile information is used to drive several traditional optimizations, as well as a novel algorithm for performing feedback-directed control flow graph node splitting. We empirically evaluate this syst ...

**Keywords:** adaptive optimization, dynamic optimization, online algorithms, virtual machines

#### 10 In or out?: putting write barriers in their place

Stephen M Blackburn, Kathryn S McKinley

June 2002 **ACM SIGPLAN Notices , Proceedings of the third international symposium on Memory management**, Volume 38 Issue 2 supplement



Full text available:  pdf(121.39 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

In many garbage collected systems, the mutator performs a write barrier for every pointer update. Using generational garbage collectors, we study in depth three code placement options for remembered-set write barriers: inlined, out-of-line, and partially inlined (fast path inlined, slow path out-of-line). The fast path determines if the collector needs to remember the pointer update. The slow path records the pointer in a list when necessary. Efficient implementations minimize the instructions on ...

**Keywords:** Java, copying collection, generational collection, write barriers

#### 11 Optimization of custom MOS circuits by transistor sizing

Andrew R. Conn, Paula K. Coulman, Ruud A. Haring, Gregory L. Morrill, Chandu Visweswariah  
January 1997 **Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design**

Full text available:  pdf(68.85 KB)  Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)  
[Publisher Site](#)

Optimization of a circuit by transistor sizing is often a slow, tedious and iterative manual process which relies on designer intuition. Circuit simulation is carried out in the inner loop of this tuning procedure. Automating the transistor sizing process is an important step towards being able to rapidly design high-performance, custom circuits. JiffyTune is a new circuit optimization tool that automates the tuning task. Delay, rise/fall time, area and power targets are accommodated. Each (weig ...


**Keywords:** Circuits, transistor sizing, optimization, simulation, gradients.

#### 12 Cache behavior of network protocols

Erich Nahum, David Yates, Jim Kurose, Don Towsley



June 1997 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the**

**1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 25 Issue 1

Full text available:  [pdf\(1.67 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)


In this paper we present a performance study of memory reference behavior in network protocol processing, using an Internet-based protocol stack implemented in the x-kernel running in user space on a MIPS R4400-based Silicon Graphics machine. We use the protocols to drive a validated execution-driven architectural simulator of our machine. We characterize the behavior of network protocol processing, deriving statistics such as cache miss rates and percentage of time spent waiting for memo ...

- 13** [ProfileMe: hardware support for instruction-level profiling on out-of-order processors](#)  
 Jeffrey Dean, James E. Hicks, Carl A. Waldspurger, William E. Weihl, George Chrysos  
 December 1997 **Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture**


Full text available:  [pdf\(1.60 MB\)](#)  Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)  
[Publisher Site](#)

Profile data is valuable for identifying performance bottlenecks and guiding optimizations. Periodic sampling of a processor's performance monitoring hardware is an effective, unobtrusive way to obtain detailed profiles. Unfortunately, existing hardware simply counts events, such as cache misses and branch mispredictions, and cannot accurately attribute these events to instructions, especially on out-of-order machines. We propose an alternative approach, called ProfileMe, that samples instructio ...

- 14** [A new multicasting-based architecture for Internet host mobility](#)  
 Jayanth Mysore, Vaduvur Bharghavan  
 September 1997 **Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking**

Full text available:  [pdf\(2.08 MB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

- 15** [Static correlated branch prediction](#)  
 Cliff Young, Michael D. Smith  
 September 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 5

Full text available:  [pdf\(508.49 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Recent work in history-based branch prediction uses novel hardware structures to capture branch correlation and increase branch prediction accuracy. Branch correlation occurs when the outcome of a conditional branch can be accurately predicted by observing the outcomes of previously executed branches in the dynamic instruction stream. In this article, we show how to instrument a program so that it is practical to collect run-time statistics that indicate where branch correl ...

**Keywords:** branch correlation, branch prediction, path profiling, profile-driven optimization

- 16** [On hot-spot contention in interconnection networks](#)  
 N. M. Patel, P. G. Harrison  
 May 1988 **Proceedings of the 1988 ACM SIGMETRICS conference on Measurement and modeling of computer systems**

Full text available:  [pdf\(770.61 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

A major component of a parallel machine is its interconnection network, which provides concurrent communication between the processing elements. It is common to use a multi-stage interconnection network (MIN) which is constructed using crossbar switches and introduces not only contention for destination addresses but also additional contention for internal switches. oth types of contention are increased when non-local communication

across a MIN becomes concentrated on a certain destination ...

**17** Procedure placement using temporal-ordering information

Nikolas Gloy, Michael D. Smith

September 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,  
Volume 21 Issue 5

Full text available:  pdf(504.56 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Instruction cache performance is important to instruction fetch efficiency and overall processor performance. The layout of an executable has a substantial effect on the cache miss rate and the instruction working set size during execution. This means that the performance of an executable can be improved by applying a code-placement algorithm that minimizes instruction cache conflicts and improves spatial locality. We describe an algorithm for procedure placement, one type of code placement ...

**Keywords:** code placement, conflict misses, temporal profiling, working-set optimization

**18** End-to-end routing behavior in the Internet

Vern Paxson

October 1997 **IEEE/ACM Transactions on Networking (TON)**, Volume 5 Issue 5

Full text available:  pdf(255.09 KB)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#), [review](#)

**Keywords:** communication system routing, computer network performance, computer network reliability, computer networks, failure analysis, internetworking, stability

**19** Cache-conscious structure definition

Trishul M. Chilimbi, Bob Davidson, James R. Larus

May 1999 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation**, Volume 34 Issue 5

Full text available:  pdf(1.30 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

A program's cache performance can be improved by changing the organization and layout of its data---even complex, pointer-based data structures. Previous techniques improved the cache performance of these structures by arranging distinct instances to increase reference locality. These techniques produced significant performance improvements, but worked best for small structures that could be packed into a cache block. This paper extends that work by concentrating on the internal organization of f ...

**Keywords:** cache-conscious definition, class splitting, field reorganization, structure splitting

**20** Using cache line coloring to perform aggressive procedure inlining

Hakan Aydin, David Kaeli

March 2000 **ACM SIGARCH Computer Architecture News**, Volume 28 Issue 1

Full text available:  pdf(701.54 KB)

Additional Information: [full citation](#), [abstract](#), [index terms](#)

Memory hierarchy performance has always been an important issue in computer architecture design. The likelihood of a bottleneck in the memory hierarchy is increasing, as improvements in microprocessor performance continue to outpace those made in the memory system. As a result, effective utilization of cache memories is essential in today's architectures. The nature of procedural software poses visibility problems when attempting to perform program optimization. One approach to increasing visibility ...

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)